WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences

Search Results -

Term	Documents
FLOATING.USPT.	98741
FLOATINGS.USPT.	48
POINT.USPT.	1304926
POINTS.USPT.	546232
(7 AND (FLOATING ADJ POINT)).USPT.	7

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Refine Search: `17 and (floating adj point)` Clear

Search History

Today's Date: 11/8/2001

DB Name	Query	Hit Count	Set Name
USPT	17 and (floating adj point)	7	L8
USPT	(plurality with functional with units) and (VLIW) and (double adj precision)	7	L7
USPT	15 and (double adj precision)	1	L6
USPT	3833904.pn.	1	L5
USPT	(second with operand$ with address$ with increment$)	67	L4
USPT	(second with operand$ with addrress$ with increment$)	0	L3
USPT	(second with operand$ with increment$)	134	L2
USPT	(operand$ with next with increment$)	78	L1

L8: Entry 1 of 7 File: USPT Oct 30, 2001

DOCUMENT-IDENTIFIER: US 6311261 B1
TITLE: Apparatus and method for improving superscalar processors

BSPR:
These and other objects, advantages, and features are provided by a processor, comprising: a fetch and decode unit to receive a first information from at least one storage device; a plurality of functional units connected to the fetch and decoder unit by a global bus; at least one register unit having a modified reorder buffer connected to a corresponding at least one register file by a retire bus, the at least one register unit connected to the plurality of functional units by a local bus, wherein the modified reorder buffer retires a second information related to the first information to the at least one register file over the retire bus, and the at least one register unit receives a third information related to the first and second information from the plurality of functional units.

BSPR:
These and other objects, advantages, and features are provided by a processor, comprising: a fetch and decode unit for fetching and decoding information; a plurality of functional units connected to the fetch and decode unit by a global bus; and a plurality of distributed instruction queues, each of the plurality of distributed instruction queues located in a corresponding one of the functional units for receiving instructions from the fetch and decode unit over the global bus.

BSPL:
To improve performance (reduce execution time), it is necessary to reduce one or more factors. The obvious one to reduce is Clock_Period, by means of semiconductor/VLSI technology improvements such as device scaling, faster circuit structures, better routing techniques, etc. A second approach to performance improvement is architecture design. CISC and VLIW architectures take the approach of reducing Inst_Count. RISC and superscalar architectures attempt to reduce the CPI. Superpipelined architectures increase the degree of pipelining to reduce the Clock_Period.

DRPR:
FIG. 2 shows a register mapping table in the IBM RS/6000 floating point unit.

DRPR:
FIG. 33 illustrates a floating-point arithmetic logic unit (FALU).

DEPR:
The floating-point unit of the IBM RS/6000 (Trademark of IBM, Inc.) uses a 32-entry, 6-bit-wide register mapping table to implement register renaming, as shown in FIG. 2. There are 32 architectural registers and 40 physical registers. Some parts of the register-renaming structure are intentionally omitted from the original diagram to focus on the register mapping table. This register-renaming structure is implemented as a solution to the out-of-order completion problem of floating-point load/store and floating-point arithmetic instructions. In the IBM RS/6000, floating-point load/store instructions are performed independently at the fixed-point unit, which involve load/store address calculations. Without renaming, out-of-order completion can violate anti or output dependency. For example, in a floating-point load operation, the load data may return too early and overwrite a register that is still needed (i.e., it has not been read by an earlier floating-point arithmetic instruction).

DEPR:
The register renaming process is done as follows. For notational purpose, MP(i)=j (the contents of the mapping table at address i is j) indicates that architectural register Ri maps to physical register Rj. Initially, the mapping table (MP) is

reset to identity mapping, MP(i)=i for i=0, . . . , 31. A remapping is performed for every floating-point load/store instruction decoded. Suppose a floating-point load to architectural register 3, FLD R3, arrives at MP. First, the old entry of MP(3), i.e., index 3, is pushed onto the pending-target return queue (PTRQ). Then, a new physical register index from the free list (FL), index 32, is entered to MP(3). This means R3 is now remapped to R32. Further instruction codes with source operand R3 will automatically be changed to R32. Index 3 in the PTRQ is returned to FL (for reuse) only when the last arithmetic or store instruction referencing R3, prior to the FLD R3 instruction, has been performed. This ensures that the current value in physical register R3 is not overwritten while still being used or referenced.

DEPR:
Second, the mapping table is not a small circuit. For instance, the MIPS R10000 requires two 32.times.6 mapping tables, one with 12 read ports and 4 write ports for the integer register map, and another with 16 read ports and 4 write ports for the floating-point register map.

DEPR:
The dispatch stack (DS) is a central instruction window that performs dynamic code scheduling on the dynamic instruction stream of multiple functional unit processors. It allows out-of-order, multi-instruction issue. The instruction window behaves like a stack where instructions are allocated at the top and issued from the bottom. After a set of instructions is issued, the gaps (freed entries) are filled with unissued instructions above it (compression). Then, the next set of instructions can be pushed in. This is important to determine instruction order during dependency checks. A DS entry consists of an instruction tag, opcode, source and destination register identifiers, dependence fields, and issue index. To explain how the DS works, consider the program example shown below (see R. D. Acosta, J. Kjelstrup, and H. C. Torng, "An Instruction Issuing Approach to Enhancing Performance in Multiple Functional Unit Processors," IEEE Transactions on Computers, Vol. C-35, pp. 815-828, 1986), which adds floating-point numbers in R0 through R7 and leaves the sum in R0.

DEPR:
In each clock cycle, the RUU performs four operations simultaneously: (a) dispatch/allocate one instruction from the decoder, (b) issue one instruction nearest to the head pointer with ready operands, (c) writeback any result value to the instruction's RUU entry, and forward this result to any matching operand(s), and (d) retire one completed instruction at the head entry and update its result value to the register file. To describe these operations, consider the previous program listing. FIG. 5(a) shows the instruction timing when each instruction is allocated, issued, written back, and retired. It is assumed that each floating-point add takes six cycles to complete. FIG. 5(b) shows the snapshot of the RUU contents at cycle 7. Instruction I6 (fadd R0,R4,R0) has just been allocated at the tail of the RUU. Its program counter, functional unit source, operands, and destination register tag are written. The destination register tag (0,3) is simply the register number (0) appended with the current LI counter value for R0 (3). The "executed" flag is reset to indicate entry 6 as unissued. Operands are read directly from the register file. If an operand is available, its value is copied to the allocated entry and the "ready" flag is set. However, if an operand has not been produced, then its register tag is copied to the allocated entry and the "ready" flag is reset. Later when the operand value is produced, the RUU forwards it. By copying operands and forwarding results to the instruction window, anti and output dependencies are effectively eliminated (see Johnson, 1991).

DEPR:
There are four stages/operations in the Metaflow architecture involving the DRIS: dispatch/allocate, issue, writeback, and retire. (To be consistent with the terms used in this document, the Metaflow's terms of "issue", "schedule", and "update" have been changed to the similar terms dispatch/allocate, issue, and writeback, respectively.) To describe these operations, consider the previous program example. FIG. 7(a) shows the instruction timing when each instruction is allocated, issued, written back, and retired. It is assumed that there are four allocate ports, four retire ports, and two floating-point adders with 3-cycle latency. FIGS. 7(b) and 7(c) show the DRIS contents at different time points.

DEPR:
Tomasulo introduced reservation stations in the floating-point section of the IBM

360/91 to exploit the multiple execution units. The main objective was to permit simultaneous execution of independent instructions while preserving the precedence (data dependency) constraints in the instruction stream. Reservation stations are essentially an implementation of distributed instruction windows with some result shelving. The result shelving, done by copying operands and result forwarding to reservation stations, are necessary to support register renaming. FIG. 8 shows the hardware implementation of Tomasulo's algorithm. (To focus the discussion on the reservation stations, two hardware units (floating point buffers and store data buffers) are intentionally omitted from the original diagram.)

DEPR:
There are four key components in Tomasulo's concept: busy bit, tag, reservation station, and common data bus. A busy bit is associated with each floating-point register or operand as a dependency mechanism. If set (busy bit=1) then it means the register is not available, currently being generated. A tag is associated with each register instance, which is used in place of the register number/identifier. This reintroduces the distinction between register and value, the essence of register renaming. In Tomasulo' algorithm, a tag corresponds directly (1-to-1) to a reservation station. For example, in the IBM 360/91, tag numbers 10 through 12 correspond to the three reservation stations of the adder unit. However, Weiss and Smith suggested a more flexible way of assigning tags (S. Weiss and J. E. Smith, "Instruction Issue Logic in Pipelined Supercomputers," IERE Transactions on Computers, Vol. C-33, No. 11, pp. 1013-1022, 1984). When an instruction is decoded, a new tag is assigned for its destination register from a "tag pool" that consists of some finite set of tags. When the instruction completes, the tag is returned to the pool for reuse.

DEPR:
The actions taken during instruction decode are as follows. The decoder decodes one instruction from the top of the floating point operation stack (FLOS). A reservation station is allocated at the appropriate execution unit. Instruction operands (sink and source) are copied from FLR to the reservation stations. If the busy bit of an operand register in FLR is clear (indicating the register value is valid), then the register content, tag, and busy bit are copied to the reservation station (at sink or source fields). A new tag is updated to the destination register in FLR, and the busy bit is set. This new tag is the reservation station's assigned number. However, if the busy bit of the operand register in FLR is already set (indicating another instruction is currently generating the register value), then only the register tag and busy bit are copied to the reservation station.

DEPR:
The IBM RS/6000 is a multi-chip superscalar processor with a RISC instruction set (derivation of the 801 instruction set), capable of executing up to four instructions per cycle. FIG. 9 shows the architecture of the IBM RS/6000. There are three functional units--the branch unit (BU), fixed-point unit (FXU), and floating-point unit (FPU)--that are capable of executing instructions in parallel. The BU can issue up to two instructions per cycle, a branch and a condition-register instruction. The FXU issues one instruction per cycle, which can be a fixed-point arithmetic, a fixed-point load/store, or a floating-point load/store. The FPU issues one floating-point arithmetic instruction per cycle including a multiply-add instruction. Each functional unit has instruction buffers (I-buffers) to shelve instructions. These I-buffers are organized as a FIFO instruction window with in-order issue. The BU's I-buffers are a central window that contain all fetched instructions (not decoded yet). The FXU's and FPU's I-buffers are distributed windows. Both receive the same fixed- and floating-point instructions (not decoded at this point).

DEPR:
The BU receives four instructions per cycle from the I-cache arrays into the BU's 12-entry I-buffers and dispatch unit. The dispatch unit dispatches externally to the FXU and/or FPU any two-instruction combination of fixed- and floating-point instructions. If the remaining two instructions contain a branch and/or a condition-register instruction, they are dispatched and executed internally in the BU. When a conditional branch instruction is encountered, the BU fetches the branch-not-taken path instructions (default branch prediction direction) and dispatches them to the FXU and FPU. These instructions are executed speculatively and can be canceled if the branch is mispredicted (by postponing retirement to register files and flushing the pipelines). The branch-taken path instructions are

also fetched from the I-cache arrays and placed at the BU's I-buffers, but their dispatching (or flushing) is held off until the branch outcome is known. The worst-case penalty for a branch misprediction is three cycles. The penalty can be eliminated if there are enough independent instructions to separate the compare from the branch.

DEPR:
The BU has four special purpose registers (see R. R. Oehler and R. D. Groves, "The IBM RISC/6000 Processor Architecture," IBM Journal of Research and Development, Vol. 34, No. 1, pp. 23-36, 1990); the machine-state register (to control system states), the link register (for subroutine return address), the count register (to control loop iteration), and the condition register (to support multiple condition codes for conditional branches). Zero-delay branch for loops with a known iteration count is achieved with the branch-and-count instruction that uses the count register. The condition register contains eight condition fields, two of which are reserved to contain the condition code results of arithmetic computations in the FXU and FPU. The remaining six can be explicitly set by other fixed- or floating-point compare instructions and special branch-unit instructions. The setting of each condition field is controlled by the record bit (Rc) of arithmetic instructions. There are advantages to having multiple, setable condition fields. First, the compiler can schedule a compare instruction early, as far away as possible from the conditional branch instruction. Second, several compare instructions can be scheduled first (their results written into separate condition fields), which are then followed by a single conditional branch instruction. This is useful to implement a guarded statement/code section with multiple guard conditions, eliminating the typical structure of a series of single compares followed by a single branch.

DEPR:
The FXU contains I-buffers, an arithmetic logic unit (ALU), a general-purpose fixed-point register file, and a single-entry store buffer. The I-buffers receive both fixed- and floating-point instructions from the BU's dispatch unit, but issue only fixed-point instructions and floating-point load/store instructions to the ALU. Addresses of all loads/stores are computed in the FXU. The ALU includes a multiply/divide unit with 3- to 5-cycle multiply and 19- to 20-cycle divide latencies. The store buffer holds data and address of one fixed-point store instruction. The store buffer makes load bypassing possible. Address and data of floating-point store instructions are buffered in the FPU. The I-buffer is a strictly FIFO instruction window with in-order issue. Partial out-of-order issue is achieved among different functional units. Since there is only one ALU and no out-of-order issue in the FXU, the integer RF is not accompanied by a result buffer. Result values are written directly to the integer RF, except on speculative results which are held off in the pipeline until the branch condition is known. Further instruction issue/execution in the FXU must be stalled. This limits the speculative execution capability in the IBM RS/6000.

DEPR:
The FPU contains I-buffers, a unified floating-point multiply-add-fused unit (MAF), a floating-point register file, a register-mapping table, and a store buffer. FPU's I-buffers receive the same instructions as FXU's I-buffers, but issue only floating-point arithmetic instructions to the MAF. The MAF can perform an indivisible multiply-accumulate operation (A.times.B)+C, which reduces the latency for chained multiply-add operations, rounding errors, chip busing, and the number of adders/normalizers. The latency of a floating-point multiply-add instruction (FMA) is two cycles (see R. K. Montoye, et al., "Design of the IBM RISC System/6000 Floating-Point Execution Unit," IBM Journal of Research and Development, Vol. 34, No. 1, pp. 59-70, 1990). The register-mapping table provides register renaming (8 renaming registers) to allow independent, out-of-order executions of floating-point load/store and arithmetic instructions. The store buffer contains five entries for addresses and four entries for data. A floating-point store instruction is issued at the FXU where the store address is calculated and placed at the store buffer. Once the FPU produces the store value, it is placed at the corresponding entry in the store buffer, ready to be committed to the data cache (it is to be understood that memory or other storage units may be used instead of a data cache). By buffering stores, the FXU can continue issuing subsequent loads (load bypassing).

DEPR:
FIG. 10 shows the cycle-by-cycle execution of the inner loop. The superscripts

indicate the iteration numbers. During cycle 1, four instructions (I7.sup.0, I8.sup.0, I9.sup.0, I10.sup.0) starting from_LOOP label are fetched from the I-cache arrays and placed into BU's I-buffers. During cycle 2, the first load and multiply-add instructions (17.sup.0, I8.sup.0) are sent to the FXU and FPU, respectively. The next four instructions are fetched (I11.sup.0, I12.sup.0, I13.sup.0, I14.sup.0). During cycle 3, the FXU decodes the floating-point load (I7.sup.0) and discards the floating-point multiply-add (18.sup.0). The FPU pre-decodes both instructions for register renaming. The loop-closing BCT instruction (I15.sup.0) is fetched. During cycle 4, there is no valid instruction fetch because the branch target address is still being computed. The FXU computes the address of the first load (I7.sup.0), while decoding the second load (I9.sup.0). The FPU renames the floating-point registers of I7.sup.0 and I8.sup.0. The BU detects the BCT instruction and generates the branch target address. During cycle 5, instructions from the next iteration (I7.sup.1, I8.sup.1, I9.sup.1, I10.sup.1) are fetched. The D-cache is accessed for the first load (I7.sup.0). The FXU computes the address of the second load (I9.sup.0). The first FMA instruction (I8.sup.0) is decoded at the FPU. During cycle 6, the FPU executes the first FMA instruction while decoding the second FMA instruction (I10.sup.0). The D-cache is read for the second load (I9.sup.0). In summary, the first iteration outputs x(i)' and y(i)' are stored at cycle 10 and 11, respectively. The iteration period of this loop is 4 cycles. In FIG. 10, there is no branch penalty (zero-cycle branch) in FXU's and FPU's pipelines. The execute pipeline stages (FXE, FPE1, FPE2) are always full, primarily because the instruction fetch rate is twice the instruction issue rate at the arithmetic units. However, a true branch penalty should be seen at the fetch stage (IF), which in this case shows a one-cycle branch delay due to the branch address calculation.

DEPR:
The IBM RS/6000 processor chipset consists of nine chips (including I-cache and D-cache arrays, bus controller, and I/O controller), which are implemented in a 1-.mu.m, three-metal CMOS process. The total number of transistors is 6.9 million. The benchmark performance figures on the top of the line system, the IBM RS/6000 POWERSTATION (Trademark of IBM Corporation) 580 (62.5 MHz), are SPECint92 61.7, SPECfp92 133.2 (see R. Myrvaagnes, "Beyond Workstations," Electronic Products, pp. 17-18, 1993), and 38.1 MFLOPS on (double precision, N=100) LINPACK (Trademark of Digital Equipment Corporation--see Digital Equipment Corporation, ALPHA AXP Workstation Family Performance Brief-Open VMS, 1992).

DEPR:
In general, the IBM RS/6000 supports all six superscalar features (multi-instruction issue, decoupled dataflow scheduling, out-of-order execution, register renaming, speculative execution, and precise interrupts), some in a limited way. Although four instructions are fetched per cycle, only two (fixed- and floating-point) or three (including a branch) instructions are typically issued per cycle. Only single-level speculative execution is supported. Multiple unresolved conditional branches cause issue stalls because of the lack of a result buffer. Precise interrupts are not supported in the regular mode. They are only supported when the processor is put in the "synchronize" mode, which slows the processor significantly.

DEPR:
The LIGHTNING SPARC microprocessor, from Metaflow Technologies, Inc., is the first implementation of the Metaflow architecture that executes the SPARC (v.8) RISC instruction set. The architecture is based on the DCAF (dataflow content-addressable FIFO), a DRIS implementation. Although the DRIS is conceptually a central window, it is implemented as three physical windows: (a) the central DCAF in DIU which shelves all instructions (complete DRIS), (b) the floating-point DCAF in FPU which shelves only floating-point instructions, and (c) the branch DCAF in IIU which shelves only conditional branch instructions. The central DCAF is the central window that is responsible for retiring operations; while others are only a subset of the central DCAF. FIG. 11 shows the LIGHTNING SPARC module which consists of an external cache (up to 1 Mbyte) and four ASICs; the instruction issue unit (IIU), the dataflow integer unit (DIU), the floating-point unit (FPU), and the cache controller/MMU/bus interface (CMB). The external cache consists of the first-level cache for data and the second-level cache for instructions (the first-level is in the IIU chip).

DEPR:
The IIU's branch unit executes a conditional branch instruction speculatively, and

shelves it at the branch DCAF for a later misprediction check. The branch DCAF is a specialized DRIS implementation that shelves conditional branch instructions that were speculatively executed. The ID of the oldest, unresolved conditional branch instruction is sent to the retire logic in the central DCAF, to prevent retiring speculative instructions. During the writeback stage, the IDs and results of up to three condition code-setting instructions (two from the integer ALUs in DIU and one from the floating-point adder in FPU) are updated to the branch DCAF. During the execute stage, all branch entries with unlocked operands compare their condition code result with the prediction. If not the same, then the conditional branch was misspeculated. Branch repair is initiated on the oldest mispredicted branch. Its ID is broadcast to all DCAFs. All entries with younger IDs are flushed by moving the tail pointers accordingly.

DEPR:
The DIU contains the central DCAF, two integer ALUs, one memory-address ALU, and retire logic. Up to three instructions can be allocated into the central DCAF, including floating-point instructions which are used for retirement purposes. The central DCAF invokes issue and writeback operations only on integer and memory-reference instructions. To allow proper retiring, each cycle the central DCAF is informed by the FPU on all IDs of successfully executed floating-point instructions. The retire logic can retire up to eight instructions per cycle in the central DCAF (see Popescu, et al.): three that update integer registers or condition codes, two that update floating-point registers, one store instruction, and any two instructions of other types (control transfers, processor state updates). The FPU contains the floating-point DCAF, a floating-point adder, and a floating-point multiplier. The floating-point DCAF invokes allocate, issue, and writeback only on floating-point instructions. To deallocate entries, the floating-point DCAF is informed by the DIU on the IDs of retired floating-point instructions.

DEPR:
The SUPERSPARC processor from Texas Instruments, Inc. is the first commercial superscalar implementation of the SPARC version 8 architecture (Sum Microsystems Computer Corporation, The SUPERSPARC Microprocessor-Technical White Paper, 1992). A virtually identical version from Ross Technology, Inc. and Cypress Semiconductor Corporation is called the HYPERSPARC (Trademark of Ross Technology, Inc. and Cypress Semiconductor Corporation). FIG. 12 shows the SUPERSPARC architecture which primarily consists of three functional units: an instruction unit, integer unit, and floating-point unit. There are also a 20-Kbyte I-cache that fetches four instructions per cycle, and a 16-Kbyte D-cache that can handle one 64-bit load or store per cycle. These on-chip caches can interact with the MBus or a second-level cache controller that supports up to 1-Mbyte of external cache.

DEPR:
The floating-point unit provides a 4-entry floating-point instruction queue, 5-port floating-point register file, floating-point adder, and floating-point multiplier. A floating-point instruction is issued from the bottom (oldest entry) of the instruction queue when the operands and resources are available. All floating-point instructions start in order and complete in order (see Sun Microsystems Computer Corporation, 1992). The floating-point adder performs addition, subtraction, format conversion, comparison, absolute value, and negation. The floating-point multiplier performs single- and double-precision multiplication, division, square root, and integer multiplication and division. Bypassing capabilities from the result buses and load bus to arithmetic units are provided. The latency of most floating-point operations is three cycles.

DEPR:
(3) Only limited out-of-order execution is supported; no load bypassing, and strictly in-order issue with the possibility of out-of-order completion of floating-point instructions from the floating-point queue with respect integer instructions.

DEPR:
The DEC ALPHA 21064 processor is the first implementation of Digital Equipment Corporation's 64-bit ALPHA AXP architecture (see E. McLellan (Digital Equipment Corporation), "The APLHA AXP Architecture and 21064 Processor," IEEE Micro, pp. 36-47, 1993). It is currently the fastest single-chip microprocessor in the industry. The architecture is a combination of superpipelined and superscalar architectures. The integer and floating-point pipelines are seven- and ten-stages

deep, respectively. Since DEC has an existing, large customer base of software, it offers compatibility with VAX and MIPS codes through binary translation. Executable program codes are converted to AXP code without recompilation (with some performance degradation). FIG. 13 shows the DEC ALPHA 21064 architecture, which has four functional units: an instruction unit (IBox), an integer unit (EBox), a floating-point unit (FBox), and an address unit (ABox). There are also 32 entry by 64-bit integer and floating-point register files (RFs), 8-Kbyte D-cache, and 8-Kbyte I-cache with a 2 K by 1-bit branch history table. The branch history table is provided for dynamic prediction and achieves 80% accuracy on most programs. Static prediction is also supported based on the sign of the branch address displacement field as the default; backward branches are predicted taken and forward branches are predicted not-taken.

DEPR:
The EBox contains dedicated integer multiplier, adder, shifter, and logic units. The multiplier unit is not pipelined to save silicon area. The adder and logic units have single-cycle latency with bypass paths for register write data. The shifter takes two cycles to produce results, but is fully pipelined (one-cycle throughput). The FBox contains dedicated floating-point multiplier/adder and divider units. It supports both VAX- and IEEE-standard data types and rounding modes. The divider unit generates one bit of quotient per cycle. All other floating-point operate instructions have six-cycle latency and one-cycle throughput.

DEPR:
FIG. 14 shows the pipeline stages of the DEC ALPHA 21064 processor for integer and floating-point instructions. Up to two instructions can be processed in each stage. The first three stages (IF, SW, I0) can be stalled, while stages beyond I0 advance every cycle (see D. W. Dobberpuhl, et al., "A 200-MHz 64-Bit Dual-Issue CMOS Microprocessor," Digital Technical Journal, Vol. 4, No. 4, Special Issue, pp. 35-50, 1992). In stage IF, a pair of instructions is fetched from the on-chip I-cache. In stage SW, a swap or serialization operation is performed based on the dual-issue rules. If there is a conditional branch instruction, the branch direction is predicted statically or dynamically (using the branch history table). In stage I0, instruction(s) are decoded and checked for dependencies between the two fetched instructions (if any). In stage I1, instruction(s) are issued to the appropriate functional unit, provided there is no register conflict. The source operands are read from the integer and/or floating-point RFs and sent to the EBox, IBox, ABox, and FBox. In stage 4, instruction executions start (stage A1 for integer instructions, stage F1 for floating-point instructions).

DEPR:
FIG. 14(a) shows that all integer arithmetic and logic instructions (EBox), except shift instructions, have one-cycle latency, through bypass paths. Shift instructions have two-cycle latency. All results in EBox are actually written back to the integer RF in stage 6. Without the bypass path, the latency would be three cycles. But with the bypass path, the latency is reduced to one or two cycles. This improves the probability that back-to-back dependent instructions execute at full pipeline speed. The DECchip 21064 dedicates 45 different bypass paths. Conditional branch instructions (IBox) are resolved in stage 4. If a branch misprediction is detected, a branch repair is initiated. Instructions subsequent to the branch (in the wrong path) and their intermediate results are flushed from all pipeline stages. The alternate branch target address is computed as the new PC. The first instruction pair of the correct branch path is fetched at stage 6. This branch misprediction causes a four-cycle delay. Primary, on-chip D-cache accesses of load and store instructions (ABox) complete in stage 6. So, the latency of loads and stores is three cycles. FIG. 14(b) shows that results of floating-point operations (from the multiplier/adder unit) are written back to the floating-point RF in stage 9, thus giving a 6-cycle latency. The ALPHA AXP architecture has several notable characteristics:

DEPR:
The DEC ALPHA 21064 single-chip processor is implemented using a 0.75 .mu.m, three-metal CMOS process, with operating speeds up to 200 MHz. The extremely high clock frequency presents a difficult clocking situation. To avoid race conditions for latched data, the clock edge rate must be extremely fast (0.5 ns) and only very little clock skew can be tolerated. DEC's solution is to implement a very large, on-chip clock driver with a final stage containing 156 to 172-mil-wide PMOS and 63 to 78-mil-wide NMOS devices (see McLellan). The clock driver occupies about

5% of the total chip area and draws a peak switching current of 43 A. A 0.13-.mu.F on-chip decoupling capacitance must be added to overcome the supply voltage problem. The chip's power dissipation is 30 W at 200 MHz with a 3.3-V supply. Sophisticated packaging is used to cool the chip. These hardware cost and implementation problems are compensated by top performance. The benchmark performance figures on the top-of-the-line system, the DEC 10000/610 (200 MHz), are: SPECint92 116.5, SPECfp92 193.6, and 40.5 MFLOPS on 100.times.100 Linpack (double precision).

DEPR:
The HP PA-7100 processor is the seventh implementation of Hewlett-Packard's PA-RISC (precision architecture, reduced instruction set computer--Trademark of Hewlett-Packard, Inc.) architecture (T. Asprey, et al. (Hewlett-Packard), "Performance Features of the PA-7100 Microprocessor," IEEE Micro, pp. 22-35, 1993). It is the first superscalar PA-RISC design, which issues up to two instructions per cycle. Its design also has a VLIW flavor. There are two notable design approaches in the PA-RISC architecture; (a) the use of off-chip, rather than on-chip, primary caches (I-cache and D-cache), and (b) the reduction of instruction count in programs (pathlength reduction--see R. Lee, et al., "Pathlength Reduction Features in the PA-RISC Architecture," Proceedings of the 37th COMPCON, pp. 129-135, 1992) by adding VLIW-like and SIMD-like instructions. The motivation to use off-chip caches is the fact that on-chip caches are usually not large enough to achieve balanced performance across a wide range of applications. Typically, on-chip I-caches range from 8 to 20 Kbytes, and on-chip D-caches range from 8 to 16 Kbytes. The PA-7100 processor can have up to 1 Mbyte I-cache and 2 Mbyte D-cache. Unlike most processors with small on-chip caches, a secondary cache becomes unnecessary. Another advantage is the flexibility of cache size and speed to configure different systems, from low-end to high-end systems.

DEPR:
The objective of pathlength reduction is to resolve the key disadvantage of RISC architectures, the code/pathlength expansion. There are two instruction types added to the RISC instruction set. First, two or three operations that frequently occur together are combined into a fixed-length, 32-bit instruction. This results in multi-operation, VLIW-like instructions (except they are contained within a short 32-bit instruction), such as Shift&Add (perform integer multiplications with a small constant), Multiply&Add (floating-point), Compare&Branch, Add&Branch, Branch_on_Bit, etc (see Lee, et al.). Other streamlined RISC architectures such as MIPS require multiple instructions to perform these tasks. Second, SIMD-like instructions are added to operate, in parallel, on multiple data units smaller than a 32-bit word. These instructions are particularly useful in parallel character and decimal operations. For example, in the C language, character manipulations frequently involve finding the null byte (zero) that marks the end of a variable-length string of characters. PA-RISC's Unit_Exclusive_Or instruction speeds this process by testing a "no byte zero" in a word of four bytes in a single cycle (see Lee, et al.). The addition of the two instruction types is accommodated in the hardware without impacting the cycle time or the CPI. This gives the PA-RISC architecture some of the advantages of a very simple VLIW architecture (with short 32-bit instructions), without losing the advantages of a RISC architecture.

DEPR:
FIG. 15 shows the PA-7100 architecture. The processor chip consists of six major blocks; the integer unit, floating-point unit, cache control/interface, unified TLB, control unit, and system bus interface. The control unit is responsible for fetching, decoding, and issuing of instructions. Two instructions are fetched from the off-chip I-cache per cycle, and buffered in a small prefetch buffer (central window). The control unit can issue up to two instructions per cycle, one to the integer unit and one to the floating-point unit. There are no alignment or order constraints on the pair of instructions (see E. DeLano, et al., "A High Speed Superscalar PA-RISC Processor," Proceedings of the 37th COMPCON, pp. 116-121, 1992). However, no two integer or floating-point instructions can be issued simultaneously. If a conditional branch instruction is encountered, a simple static branch prediction scheme is used to minimize branch penalty. All forward conditional branches are untaken and backward conditional branches are taken.

DEPR:
The integer unit contains an ALU, shift-merge unit (SMU), dedicated branch adder, and a 32.times.32-bit, general-purpose, integer register file. Besides integer

arithmetic instructions, the integer unit also executes branch instructions, loads and stores of integer and floating-point registers, and all VLIW-like and SIMD-like instructions, except the floating-point Multiply&Add and Multiply&Subtract. The VLIW-like instructions improve the utilization of the three hardware units. For example, the Add&Branch uses the ALU and the branch adder simultaneously, while the Branch_on_Bit uses the SMU and the branch adder (see Lee, et al.). The register bypass paths produce a one-cycle latency for integer arithmetic instructions.

DEPR:
The floating-point unit contains a floating-point ALU (FALU), multiplier (FMUL), divide/square root unit (FDIV/SQRT), and a 32.times.64-bit floating-point register file. Although there are 32 physical registers, the first four registers (0-3) are dedicated for status register and exception registers. The remaining 28 registers (4-31) are used as register operands for arithmetic operations. Each register can be access as a 64-bit double word or as two 32-bit single words. The FALU performs single- and double-precision add/subtract, compare/complement, and format conversion instructions. The FMUL performs single- and double-precision multiplications, and also 32-bit unsigned integer multiplications (64-bit result). The multiplier array is based on a radix-4 Booth encoding algorithm. The register bypass paths produce a two-cycle latency for all floating-point instructions performed in the FALU and FMUL. The FDIV/SQRT performs floating-point divide and square-root operations based on a modified radix-4 SRT (Sweeney, Robertson, and Tocher--see Asprey, et al.) algorithm. The main modification is running the radix-4 division hardware at twice the processor clock frequency to effectively achieve a radix-16 performance. Four quotient bits are computed each clock cycle, giving a latency of 8 and 15 cycles for single- and double-precision divide/square root operations. The floating-point register file has five read ports and three write ports to allow concurrent execution of a floating-point multiply, a floating-point add, and a floating-point load or store. This occurs when a Multiply&Add or a Multiply&Subtract instruction is issued concurrently with a floating-point load/store instruction (categorized as an integer instruction).

DEPR:
The instruction execution pipeline for various types of instructions is shown in FIG. 16. The pipeline frequency is determined by the read cycle time of the off-chip cache RAMs (see Asprey, et al.). Each pipeline stage is divided into two equal phases (2-phase clocking scheme). The first three phases are dedicated for instruction fetching from the off-chip I-cache. Instruction decode and issue can be done in a mere single phase because a pre-decoded bit is dedicated in the instruction-field format to steer instructions to the integer and floating-point units. The phases for instruction execution depend on the instruction type, as depicted in FIG. 16. For a conditional branch instruction, instructions along the predicted path are fetched (static branch prediction) while the branch condition is evaluated. In the meantime the alternate link address (Laddr) is also calculated. If at the end of the execute stage the branch is found to be mispredicted, the previous speculative instruction fetch is flushed and new instructions along the correct path are fetched. If the delay is viewed from the I-fetch to the Target I-fetch, the minimum branch delay of correctly and incorrectly predicted branches is one cycle and two cycles, respectively. The PA-7100 processor has extensive register bypass capability to minimize pipeline interlock penalties. As illustrated in FIG. 16, the penalty for integer ALU pipeline interlock is zero cycles. The penalty for load use, floating-point ALU, or floating-point multiply pipeline interlocks is one cycle.

DEPR:
The HP PA-7100 processor is implemented using a 0.8 .mu.m, three-metal CMOS process. It operates at 100 MHz and integrates about 850,000 transistors. The use of off-chip caches results in a large pin-count package, 504-pin PGA. The reported benchmark performance figures on the top-of-the-line system, the HP9000/735 (99 MHz), are: SPECint92 80.0, SPECfp92 150.6, and 40.8 MFLOPS on 100.times.100 Linpack (double precision).

DEPR:
The HP PA-7100 processor supports four superscalar features; multi-instruction issue, decoupled dataflow scheduling, out-of-order execution, and precise interrupts. However, it does not support register renaming and speculative execution. Note that static branch prediction is only used for speculative prefetch, not speculative execution. Instructions following an unresolved

conditional branch are stalled and not executed. The HP PA-7100 designers rely on aggressive VLIW-like software scheduling and chose not to push the superscalar hardware too aggressively:

DEPR:
(1) The multi-instruction issue is limited to dual issue of integer and floating-point instruction pairs. No two integer or floating-point instructions can be issued simultaneously. To increase machine parallelism, the VLIW-like and SIMD-like instructions are included, which increases the complexity of the compiler.

DEPR:
The Intel PENTIUM microprocessor is the first superscalar implementation that runs the widely-used x86 CISC instruction set. The x86 instructions use only two operands and permit combinations of register and memory operands. Thus, unlike all other commercial superscalar processors, the PENTIUM processor is not a typical register-to-register, three-address machine. Despite the complexity of CISC instructions, many of which require microcode sequencing, the PENTIUM processor manages to differentiate the "simple" (RISC-like) instructions and executes them in superscalar mode (dual-instruction issue). However, complex instructions and almost all floating-point instructions must still run in scalar mode (single-instruction issue). The superscalar execution and architectural improvements in branch prediction, cache organization, and a fully-pipelined floating-point unit result in a substantial performance improvement over its predecessor, the i486 processor. When compared with an i486 processor with identical clock frequency, the PENTIUM processor is faster by factors of roughly two and five in integer and floating-point performance, respectively (D. Alpert and D. Avnon--Intel Corporation, "Architecture of the PENTIUM Microprocessor," IEEE Micro, pp. 11-21, 1993).

DEPR:
FIG. 17 shows the PENTIUM architecture. The core execution units are two integer ALUs and a floating-point unit with dedicated adder, multiplier, and divider. The prefetch buffers fetch a cache line (256 bits) from the I-cache and performs instruction aligning. Because x86 instructions are of variable length, the prefetch buffers hold two cache lines; the line containing the instruction being decoded and the next consecutive line (see Alpert and Avnon). An instruction is decoded and issued to the appropriate functional unit (integer or floating-point) based on the instruction type. Two instructions can be decoded and issued simultaneously if they are "simple" instructions (superscalar execution). Because x86 instructions typically generate more data memory references than RISC instructions, the D-cache supports dual accesses to provide additional bandwidth and simplify compiler instruction scheduling algorithms (see Alpert and Avnon).

DEPR:
The floating-point unit consists of six functional blocks: the floating-point interface, register file, and control (FIRC), the floating-point exponent (FEXP), the floating-point multiplier (FMUL), the floating-point adder (FADD), the floating-point divider (FDIV), and the floating-point rounder (FRND). The FIRC contains a floating-point register file, interface logic, and centralized control logic. The x86 floating-point instructions treat the register file as a stack of eight registers, with the top of the stack (TOS) acting as the accumulator. They typically use one source operand in memory and the TOS register as the other source operand as well as the destination register. In the case of 64-bit memory operands, both ports of the D-cache are used. To swap the content of the TOS register with another register, the FXCH instruction (non-arithmetic floating-point instruction) is used. The FIRC also issues floating-point arithmetic instructions to the appropriate arithmetic blocks. Non arithmetic floating-point instructions are executed within the FIRC itself. Floating-point instructions cannot be paired with any other integer or floating-point instructions, except FXCH instructions.

DEPR:
The FEXP calculates the exponent and sign results of all floating-point arithmetic instructions. The FADD executes floating-point add, subtract, compare, BCD (binary coded decimal), and format conversion instructions. The FMUL executes single-, double-, extended-precision (64-bit mantissa) floating-point multiplication and integer multiplication instructions. The FDIV executes floating-point divide, remainder, and square-root instructions. And, the FRND performs the rounding

operation of results from FADD and FDIV. The floating-point unit also supports eight transcendental instructions such as sine (FSIN), cosine (FCOS), tangent (FPTAN), etc. through microcode sequences. These instructions primarily involve the FADD arithmetic block and sometimes other arithmetic blocks.

DEPR:
The PENTIUM processor is implemented using a 0.8 .mu.m BiCMOS process. It integrates 3.1 million transistors and currently runs at 66 MHz. The integer pipeline consists of five stages: prefetch (PF), first decode (D1), second decode (D2), execute (E), and writeback (WB). The floating-point pipeline consists of eight stages, where the first three stages (FP, D1, and D2) are processed with the resources in the integer pipeline. The other floating-point stages are: operand fetch (E), first execute (X1), second execute (X2), write float (WF), and error reporting (ER). The reported benchmark performance figures of the PENTIUM processor are: SPECint92 64.5 and SPECfp92 56.9.

DEPR:
Output dependencies (artificial dependencies) stall instruction issue because register renaming is not supported. Floating-point instructions also cannot be paired with any other instructions, except occasionally with FXCH instructions (but FXCH may be considered as a useless or unnecessary instruction in true register-to-register architectures). The multiple floating-point arithmetic blocks (FADD, FMUL, FDIV) are underutilized by the limitation of one floating-point instruction per cycle.

DEPR:
(4) The out-of-order execution is limited to in-order issue with the possibility of out-of-order completion between instructions in the integer and floating-point pipelines. Load bypassing is also not supported.

DEPR:
Table 4 is a summary of upcoming commercial superscalar microprocessors in 1995/1996. DEC continues to lead the pack with its new ALPHA 21164 design. The major architectural improvements from its ALPHA 21064 predecessor are quad-issue, additional integer and floating-point execution units (total 2 each), and the inclusion of a secondary cache on chip (see L. Gwennap, "Digital Leads the Pack with 21164," Microprocessor Report, Vol. 8, No. 12, pp. 1 and 6-10, Sep. 12, 1994). The last feature is the first in the history of microprocessors and makes the ALPHA 21164 the densest with 9.3 million transistors. Sun Microsystems' ULTRASPARC (Trademark of Sun Microsystems Computer Corporation) incorporates nine independent execution units, including dedicated graphics add and multiply units. The ULTRASPARC is the first implementation of the new 64-bit SPARC version 9 instruction-set architecture, which supports conditional move instructions. It also supports MPEG-2 graphics instructions in hardware to boost multimedia application performance. The IBM POWERPC 620 is the latest and currently fastest among other PowerPC models (601, 603, 604, 615). It uses reservation stations to shelve instructions at six execution units (see L. Gwennap, "620 Fills Out POWERPC Product Line," Microprocessor Report, Vol. 8, No. 14, pp. 12-17, Oct. 24, 1994). IBM put two entries for each execution unit with the exception at load/store unit (3 entries) and branch unit (4 entries). The MIPS Technologies' R10000, also known as T5, uses "decoupled architecture" (another term for decoupled dataf low scheduling) with a set of three central windows (16-entry queues) for memory, integer, and floating-point instructions (see L. Gwennap, MIPS R10000 Uses Decoupled Architecture," Microprocessor Report, Vol. 8, No. 14, pp. 17-22, Oct. 24, 1994). The R10000 uses a register-mapping table (also called rename buffer) to support register renaming. Both integer and floating-point units have 64, 64-bit physical registers that are mapped to 32 logical registers. To handle multi-level speculative execution (up to 4 conditional branches), the R10000 saves the mapping table in shadow registers when encountering a conditional branch. The HP PA-8000 is the first 64-bit PA-RISC architecture implementation (see L. Gwennap, "PA-8000 Combines Complexity and Speed," Microprocessor Report, Vol. 8, No. 15, pp. 1 and 6-9, Nov. 14, 1994). Like its predecessors (PA-7100, PA-7100), the PA-8000 will not have on-chip cache. PA-RISC is the only advanced, general-purpose microprocessor architecture that uses off-chip L1 cache. The AMD K5 is currently the fastest x86 processor, claimed to be at least 30% faster than the Intel PENTIUM at the same clock rate (on integer code--see M. Slater, "AMD's K5 Designed to Outrun PENTIUM," Microprocessor Report, Vol. 8, No. 14, pp. 1 and 6-11, 1994). Despite the CISC x86 instruction set, the architecture internally runs RISC instructions, called ROPs (RISC operations). To achieve this, x86 instructions are

predecoded as they are fetched from memory to the I-cache. The predecoder adds five bits to each byte, causing an increase of about 50% at the I-cache array. The K5 applies all the superscalar techniques that Johnson believed to be the best, the reservation station for instruction shelving and the reorder buffer for result shelving (see Johnson, 1991).

DEPR:
Assume the processor has eight execution units: a branch unit, two fixed-point ALUs, a floating-point ALU, a floating-point multiply/divide/square-root unit, two load/store units, and a fixed/floating-point move unit. The processor begins by fetching at least one instruction or multiple instructions (N.sub.d instructions in this case, which is the decoder size) from the I-cache. It is to be understood that one or more memories or other storage units may be employed instead of the I-cache for performing the same function as the I-cache. These instructions are decoded in parallel and dispatched to their respective execution unit's reservation station. For each decoded instruction, an entry is allocated at the RB to shelve its result. To read its operand(s), each operand's register number is presented to the RB and register file, in which three situations can occur. First, if there is a matched entry in the RB and the register operand value has been calculated, then the operand value is routed/copied to the instruction's reservation station. Second, if there is a match entry in the RB but the value has not finished calculation, then the operand tag is copied instead to the reservation station. Third, if there is no match entry in the RB then the value from RF is the most recent one and copied to the reservation station.

DEPR:
During the writeback stage, the execution result is written to the RB (not RF) and also forwarded to any reservation station waiting for this result value. In every cycle, each valid reservation station with unavailable operand(s) compares its operand tag(s) with all result tags to determine when to grab certain result value(s). Note that if each execution unit's output port does not have a dedicated result bus (N.sub.res <fixed-point output ports, or N.sub.resfp <floating-point output ports), then arbitration logic must be provided to resolve who can use the shared result buses at a given time.

DEPR:
Ideally we want to use the same result register tag, which is used during the result write operation, as the unique associative key for read operation. This tag is written to the RB entry during allocation. Smith and Plezkun use the RB identifier or array index as the result tag. But this tag is not unique with the presence of a second RB (e.g., for a floating-point register file). Moreover, the tag will keep changing as the FIFO queue is advanced during multi-entry retire. Tracking many different register tags plus conditional branch tags can be a nightmare. Weiss and Smith suggested a more flexible way of assigning unique result tags, which was originally proposed to be used in reservation stations (see above section entitled Reservation Stations). When an instruction is decoded, a new tag or identifier (inst_ID) is assigned from a "tag pool" that consists of some finite set of tags. Each destination register is then tagged with the inst_ID of the producer instruction. When the instruction completes, the inst_ID is returned to the pool for reuse. This tag pool, called the Instruction ID Unit (IIU), can be implemented as a circular instruction array (IA). The inst_ID is composed of (color_bit, IA_index--these are discussed in more detail below), the current "color" bit appended with its IA index (entry address), the same scheme used in the DRIS technique (see above section entitled DRIS). The color bit is used to distinguish the age or order of instructions when the valid entry area wraps around.

DEPR:
The reservation station (RS) technique, currently considered the best technique, was originally introduced by Tomasulo in 1967 in the floating-point section of the IBM 360/91 (see Tomasulo). The main objective was to permit simultaneous execution of independent instructions while preserving the precedence (data dependency) constraints in the instruction stream. Tomasulo's RS technique was essentially ahead of its time. It actually accomplishes several superscalar objectives; multi-instruction issue, decoupled dataflow scheduling, out-of-order execution, and register renaming (eliminating anti and output dependencies). Anti dependencies (write-after-read hazards) are avoided by tagging registers, copying operands to reservation stations, and forwarding results directly to reservation stations. Output dependencies (write-after-write hazards) are avoided by comparing

tags at the FLR (floating-point register unit in the IBM 360/91) on every register write, to ensure that only the most recent instruction changes the register. However, Tomasulo's algorithm lacks a mechanism to handle speculative execution. Only the most recent updates of registers are maintained, regardless of whether they are speculative updates or not. To support multi-level speculative execution, the register file can be accompanied by a reorder buffer (RB) as seen in the AMD superscalar 29 K (see Case) and K5 (see Slater), or multiple copies of register-mapping tables (RMT) as seen in IBM POWERPCs and MIPS R10000 (see Gwennap, Oct. 24, 1994).

DEPR:
FIG. 27 shows in-order issue DIQ 300 structure with N.sub.diq entry cells 395 and N.sub.d allocate ports 310, implemented as a circularly addressed register array. It has multiple allocate ports 310 and a single issue port 340. The DIQ 300 entry fields 385 vary from one execution unit to another with the first two fields, 386 and 387 (inst_ID and opcode) always present. The example shown is of a floating-point ALU which consists of: instruction tag 386 (N.sub.tag bits), opcode 387 (N.sub.opc bits), source 1 register number 388 (log.sub.2 N.sub.frf bits, N.sub.frf is floating-point RF size), source 1 register tag 389 (N.sub.tag bits), source 2 register number 390 (log.sub.2 N.sub.frf bits), and source 2 register tag 3915(N.sub.tag bits). Note that in contrast to MRB 100 which only retires field values in field 191 of appropriate cells 195 in the retire operation, DIQ 300 issues all field values of fields 386, 387, 388, 389, 390, and 391 of a single cell 395 in order during the issue operation to instruction issue register 516 (see FIG. 33). Each DIQ entry cell 395 consists of D flip-flops (DFFs) or other storage units 305 (FIG. 28) to hold these DIQ entry fields and logic for the allocate operation as determined by allocate logic in DIQ cell 395.

DEPR:
FIG. 30 shows an enhanced DIQ 400 structure of the same floating-point ALU's DIQ 300 example in FIG. 27 to allow full out-of-order issue, although a DIQ for any functional unit could be used. Note, as shown in FIG. 30 (and FIG. 27), some of fields 485 (385 in FIG. 27) vary with the type of functional unit. The enhancements in the DIQ 400 comprise the additions of: (1) "issued" 486, "RS1_rdy" 491, and "RS2_rdy" 494 flags or fields in each entry (fields 487, 488, 489, 490, 492, and 493 are identical to fields 386, 387, 388, 389, 390, and 391, respectively, of FIG. 27), (2) comparators (not shown) to match a result tag with all operand tags (RS1_tag and RS2_tag) to update their ready flags (RS1_rdy and RS2_rdy), and (3) an issue logic 475 circuitry to determine which instruction entry should be issued next.

DEPR:
In the out-of-order issue DIQ 400 structure, an entry is still allocated from the tail side. Multiple tail pointers 460 are provided to handle multiple entry allocations in entry cells 495 per cycle. A newly allocated entry has its RS1_rdy 491 and RS2_rdy 493 fields initially set based on the operand value availability at decode time. These flags are updated during the writeback stage, by forwarding result tags 420 (NOT the result values as in the RS technique) to the appropriate functional units (for example, tags of floating-point results go only to selected functional units that use floating-point operands). These result tags 420 are compared to each entry's operand tags. A match will set the corresponding ready flag (RS1_rdy 491 or RS2_rdy 493) to TRUE.

DEPR:
FIG. 31 shows superscalar processor 500 that utilizes DIQ 300 or 400 instruction shelving and MRB 100R or 100F (both have identical structure to MRB 100, but MRB 100R buffers fixed-point register values, while MRB 100F buffers floating-point register values) result shelving techniques. Processor 500 has execution units 560, 570, 580, 590, 595, 596, 505, and 506 analogous to the execution units shown in FIG. 18. Superscalar processor 500 as shown in FIG. 31 clearly reveals a significant reduction in global buses and multiplexers compared to the processor of FIG. 18. By eliminating operand value copying and result value forwarding, shared-global operand buses and both shared-global result buses and wide-bus multiplexers are avoided, replaced by private-local (module-to-module) read buses and write buses 525, respectively. The only shared-global buses left are the required instruction dispatch buses 550 to deliver decoded instructions to every execution unit's DIQ 300 or 400. In the case of out-of-order issue DIQs 400 a small number of global wires to carry result tags are added (not shown in FIG. 31).

DEPR:
In processor 500, separate register units 530 and 540 are provided for fixed-point and floating-point register results. (It is also possible to combine both types of register results in a single register unit.) With two register units, the processor 500 core area is basically segmented by Fixed-Point Register Unit (FXRU) 530 to hold general-purpose "R" registers, and the Floating-Point Register Unit (FPRU) 540 to hold floating-point "F" registers. Special purpose registers for condition codes can use any of the "R" registers, following the "no-single copy of any resource" philosophy of the DEC ALPHA architecture. A single-copy of any resource can become a point of resource contention. The "R" and "F" registers (which may be contained in either register units 530 and 540 in 515R and 515F or in MRB 100R and MRB 100F) are also used to hold fixed- and floating-point exception conditions and status/control information, respectively.

DEPR:
Fixed-point arithmetic instructions are executed in the Fixed-Point Units (FXU 0 and FXU 1) 570 and 580. Floating-point arithmetic instructions are executed in the Floating-Point Arithmetic Logic Unit (FALU) 506 and Floating-Point Multiply/Divide/Sguare-Root Unit (FMDS) 505. Note that FALU 506 also performs floating-point compare/set instructions and writes its condition code directly to FXRU 530. Conditional/Immediate Move Unit (CIMU) 596 performs register move instructions between FXRU 530 and FPRU 540, as well as within FXRU 530/FPRU 540 itself. CIMU 596 can also be dedicated to handle conditional move instructions as seen in the SPARC-64 (version 9) instruction set. Load & Store Units (LSU 0 and LSU 1) 590 and 595 perform all load and store operations and include store queues (SQ) 591 and 594, respectively, to queue the store instructions until they can be committed/executed safely, with load bypassing and two simultaneous data accesses to D-cache 511 allowed. It is to be understood that one or more memories or other storage units may be employed instead of a cache for D-cache 511 for performing an equivalent function as D-cache 511. Branch instructions and PC address calculations are executed in the Instruction Address Unit (IAU) 560. A BTB (branch target buffer), which is a combination of a branch-target address cache (or other memory or storage unit) and branch history table, is provided in IAU 560 to help eliminate some branch delays and predict branch direction dynamically. During processor 500 implementation, it is best to physically layout circuit modules/blocks such that execution units 560, 570, 580, 590, 595, 596, 505, and 506 surround their corresponding register unit 530 or 540. Execution units that access both register units 530 and 540, such as LSUs 590 and 595 and CIMU 596, can be placed between the two. In this way, local bus 525 wiring is more direct and shorter.

DEPR:
On the performance side, the good characteristics of the RS technique in achieving maximum machine parallelism have been maintained in the DIQ 300 or 400 technique. The only sacrifice made in DIQ 300 technique is the use of in-order issue with an instruction window. This may penalize performance slightly on the cycle count, which can be easily recovered through faster and simpler circuit implementation. In the end, the actual speed or performance of the processor is faster due to reduced cycle time or more operations executed per cycle. (The out-of-order issue DIQ 400 technique is at par with the RS technique in terms of cycle-count performance, but higher in terms of overall performance if the improved clock frequency is factored in.) The performance analysis confirms that a good performance speedup, on the cycle count basis, is still achieved. Based on the benchmark set used, a speedup between 2.6.times. to 3.3.times. was realized in a 4-way superscalar model over its scalar counterpart. Moreover, the performance saturates at a relatively low number of 4 DIQ 300 or 400 entries. These results can be compared to 4-way superscalar processors which typically gain less than 2.0.times. over scalar designs on the SPECint92 benchmarks (see L. Gwennap, "Architects Debate VLIW, Single Chip MP," Microprocessor Report, Vol. 8, No. 16, pp. 20-21, Dec. 5, 1994).

DEPR:
The fourth modification, read bypassing, is necessary because the most recent tag 740 of a register may still be in allocate ports 810, not yet written to RTRB 100RT. Consider the following example where ((x-y).sup.2 +z).sup.2 computation is about to take place. Assume all calculations are performed in single-precision floating point arithmetics; variables x, y, z were already loaded into register F1, F2, F3, respectively; and the result is written to register F4. Suppose the

current decode group is as follows: (N.sub.d =4)

DEPL:
Note that a branch instruction is formatted as an integer (floating-point) instruction if its condition code is in an integer (floating-point) register. The DEC ALPHA 21064 avoids condition codes, special registers, or any other single copy of a resource which can potentially become a point of contention in a multi-instruction issue environment. Compare instructions write directly to any general-purpose register (integer or floating-point, depending on the compare operation type).

DEPV:
Maximum of one floating-point arithmetic instruction,

DEPV:
Many decoded instructions have only one register operand (arithmetic instructions with immediate value operand, loads, conditional branches, floating-point convert instructions), or worse, no register operands at all (jumps, traps).

DEPW:
The AXP instruction set includes conditional move instructions for both integer and floating-point data. These instructions should help remove some conditional branches (see section below entitled Condition Move Transformation).

DETL:
Program Example for Reorder Buffer PC Instructions Comments Latency I0: 0 R2 <- 0 ;initialize loop index I1: 1 R0 <- 0 ;initialize loop count I2: 2 R5 <- 1 ;loop increment value I3: 3 R7 <- 100 ;maximum loop count I4: 4 L1:R1 <- (R2 + A) ;load A(I) 11 cycles I5: 5 R3 <- (R2 + B) ;load B(I) 11 cycles I6: 6 R4 <- R1 +.sub.f R3 ;floating-point add 6 cycles I7: 7 R0 <- R0 + R5 ;increment loop count 2 cycles I8: 8 (R0 + C) <- R4 ;store C(I) I9: 9 R2 <- R2 + R5 ;increment loop index 2 cycles I10: 10 P = L1:R0! = R7 ;cond. branch not equal

DETL:
Valid Dual Issue: Instruction A Instruction B integer operate floating-point operate integer/floating-point load integer/floating-point operate/branch integer store integer operate integer store floating-point branch floating-point store floating-point operate floating-point store integer branch integer branch integer operate floating-point branch floating-point operate

DETL:
TABLE 3 Comparisons of Commercial Superscalar Microprocessors Metaflow IBM THUNDER TI DEC HP Intel RS/6000 SPARC SUPERSPARC ALPHA 21064 PA-7100 PENTIUM Integra- Multi-chip Multi-chip Single-chip Single-chip Multi-chip Single-chip tion Clock 62.5 MHz 80 MHz 50 MHZ 200 MHZ 99 MHz 66 MHz Speed (est) SPECint92 61.7 200 (est) 68 116.5 80.0 64.5 SPECfp92 133.2 350 (est) 85 193.6 150.6 56.9 Supersca- All six All six All six, Multi-inst. All six, All six, lar Fea- except re- issue, except re- except re- tures gister decoupled gister re- gister re- renaming dataflow naming and naming sched., speculative out-of-or- execution. der exec. Instructio FIFO I-buf- DCAFs Central Central Central Central n Shelving fers (cen- (DRIS) win-dow and window window window tral, dist. (central, dist. win- (pre-fetch (prefetch (prefetch in FXU and branch, dow (for FP buf-fers) buffers) buffers) FPU) floating- inst.) point) Result Reg.-map- DCAFs None None None None Shelving ping table in FPU Indepen- 1 branch 1 branch 1 branch 1 branch 1 integer 2 ALUs, dent unit, unit, unit, unit, unit (ALU, 1 FP unit Execution 1 FX unit, 3 ALUs, 3 ALUs, 1 address shift, (add, muit, Units 1 FP unit 1 FP add, 1 address unit, branch div) (MAF) 1 FP mult add, 1 integer add), 1 FP unit unit, 1 FP unit (add, mult) 1 FP unit (mult, (add, mult, div/sqrt) div) Decode 4 instruc- 4 instruc- 4 instruc- 2 instruc- 2 instruc- 2 instruc- Size tions tions tions tions tions tions Max Issue 1 FX or FP 1 branch Triple Dual issue Dual issue Dual issue load/store inst., issue with of certain of integer of "simple" inst., 2 integer certain integer/ and floa- instruc- 1 FP arith. inst., restriction floating- ting-point tions. inst., 1 load/ s. point instruc- 1 branch store inst, operate, tions. inst., 1 FP add/ branch, and 1 condi- sub, load/store. tion-regi- 1 FP ster inst. multiply Branch Static Dynamic Static (al- Static and Static Dynamic Prediction (con-stant ways taken) dynamic (BTFN) pre-dicted- not-taken) Notes .cndot. FP mult- .cndot. Has the .cndot. Multiple- .cndot. Hybrid of .cndot. Uses off- .cndot. Supports add in 2 most com- path fetch- superpipe- chip, pri- the wide- cycles. plete dyna- ing into lined and mary caches ly-used x86 .cndot. This mic

hard- se-quential supersca- for size inst. set. RS/6000 de- ware sche- and target
lar. and speed .cndot. The only sign is the duler with inst. .cndot. True 64-
flexi- superscalar foundation full out- queues bit archi- bility processor of
of-order helps tecture. .cndot. Supports that is not follow-on issue. reduce
.cndot. Supports VLIW-like a register- single-chip .cndot. Low clock branch cond.
move and SIMD- to-regi- versions speed due mispredic- inst. like inst. ster, 3-ad-
(PowerPC to complex tion .cndot. Supports for path- dress ma- 601, 603, out-of-or-
penalty. multiple length chine. 604, 620) der issue. O/S using reduction. .cndot.
.cndot. Precise .cndot. Thunder PALcode. Inefficient interrupts SPARC was a
.cndot. Imprecise inst. only in reborn of interrupts. window due "synchroniz the
to vari- e" mode. unsuccessfu able - length l Lightning x86 inst. SPARC.